Tema 4: Resolución proposicional

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Introducción

- Problemas y soluciones
  - Problemas del automatización de deduciones:
    * Gran tamaño del espacio de búsqueda
    * "Intuición" en la elección de axioma
  - Objetivos de resolución:
    * Reducir el tamaño de los espacios de búsqueda
    * Usar una regla en lugar de MP y axiomas
- Otro problema y solución:
  - Las expresiones para la resolución tienen que estar en una forma canónica: la forma clausular
  - Hay un procedimiento de transformación de fórmulas a formas clausulares equivalentes

Equivalencia lógica

- **Fórmulas equivalentes**
  - **Def.:** F **y** G **son equivalentes syss** $\models F \leftrightarrow G$
  - **Representación:** $F \equiv G$
  - **Ejemplos:**
    $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
    $p \rightarrow q \equiv \neg p \vee q$
    $p \wedge q \equiv \neg(\neg p \vee \neg q)$
    $p \vee q \equiv \neg(\neg p \wedge \neg q)$
  - **CNS:** $F \equiv G$ **syss para toda interpretación** I, $\mathrm{sig}(F, I) = \mathrm{sig}(G, I)$
- **Propiedades de la equivalencia**
  - **Si** $F \equiv F'$, **entonces** $\neg F \equiv \neg F'$
  - **Si** $F \equiv F'$, $G \equiv G'$ **y** $\star \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$, **entonces** $F \star G \equiv F' \star G'$
  - **Sea** G **una subfórmula de** F **y** F' **la obtenida sustituyendo una ocurrencia de** G **en** F **por** G'. **Si** $G \equiv G'$, **entonces** $F \equiv F'$

Equivalencia lógica

- **Equivalencias:**
  - **Idempotencia:**
    $F \vee F \equiv F$,
    $F \wedge F \equiv F$
  - **Conmutatividad:**
    $F \vee G \equiv G \vee F$,
    $F \wedge G \equiv G \wedge F$
  - **Asociatividad:**
    $F \vee (G \vee H) \equiv (F \vee G) \vee H$,
    $F \wedge (G \wedge H) \equiv (F \wedge G) \wedge H$
  - **Distributividad:**
    $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$,
    $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$
  - **Doble negación:**
    $\neg\neg F \equiv F$
  - **Leyes de De Morgan:**
    $\neg(F \wedge G) \equiv \neg F \vee \neg G$,
    $\neg(F \vee G) \equiv \neg F \wedge \neg G$

Forma normal negativa

- Idea informal:
  - Una fórmula está en forma normal negativa si no contiene las conectivas $\rightarrow, \leftrightarrow$ y la negación no se aplica a subfórmulas compuestas
- Def. de forma normal negativa:
  - Si F es atómica, entonces F y $\neg F$ son formas normales negativas
  - Si F y G son formas normales negativas, entonces $(F \wedge G)$ y $(F \vee G)$ también lo son.
- Ejemplos de formas normales negativas:
  - $(\neg p \vee q) \wedge (\neg q \vee p)$ es forma normal negativa
  - $(p \rightarrow q) \wedge (q \rightarrow p)$ no es forma normal negativa
  - $\neg(p \wedge q)$ no es forma normal negativa

Forma normal negativa

- **Transformación a forma normal negativa:**
  - **Procedimiento FNN(F):**
    1. **Eliminación de equivalencias**
       $$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$
    2. **Eliminación de implicaciones**
       $$p \rightarrow q \equiv \neg p \vee q$$
    3. **Interiorización de negaciones**
       $$\neg(p \wedge q) \equiv \neg p \vee \neg q$$
       $$\neg(p \vee q) \equiv \neg p \wedge \neg q$$
       $$\neg\neg p \equiv p$$
  - **Ejemplos:**
    $\text{FNN}(p \leftrightarrow q) = (\neg p \vee q) \wedge (\neg q \vee p)$
    $\text{FNN}(p \vee \neg q \rightarrow r) = (\neg p \wedge q) \vee r$
    $\text{FNN}(p \wedge (q \rightarrow r) \rightarrow s) = (\neg p \vee (q \wedge \neg r)) \vee s$
- **Propiedades:**
  - **FNN(F) siempre termina**
  - **FNN(F) es una forma normal negativa**
  - **FNN(F) $\equiv F$**

Literales

- **Literales:**
  - F **es literal positivo syss** F **es fórmula atómica.**
  - $\neg F$ **es literal negativo syss** F **es fórmula atómica**
  - F **es literal syss** F **es literal positivo o negativo**
  - **Variables para literales:** $L, L_1, L_2, \ldots$
- **Complementario de un literal:**
  - $\overline{p} = \neg p$
  - $\overline{\neg p} = p$

Formas normales conjuntivas

- **Idea de forma normal conjuntiva**
  - $(L_{1,1} \vee \ldots \vee L_{1,n_1}) \wedge \ldots \wedge (L_{m,1} \vee \ldots \vee L_{m,n_m})$
- **Idea de fórmula clausular**
  - $L_1 \vee \ldots \vee L_n$
- **Definición de fórmula clausular:**
  - Si F es literal, entonces F es fórmula clausular
  - Si F y G son fórmulas clausulares, entonces $(F \vee G)$ es fórmula clausular
- **Ejemplos de fórmulas clausulares:**
  - p es una fórmula clausular
  - $\neg p$ es una fórmula clausular
  - $\neg p \vee (q \vee \neg r)$ es una fórmula clausular
  - $\neg p \vee (q \wedge \neg r)$ no es una fórmula clausular
- **Definición de forma normal conjuntiva:**
  - Si F es una fórmula clausular, entonces F es una forma normal conjuntiva
  - Si F y G son formas normales conjuntivas, entonces $(F \wedge G)$ también lo es

Formas normales conjuntivas

- **Ejemplos de formas normales conjuntivas:**
  - $(\neg p \vee q) \wedge (\neg q \vee p)$ **es forma normal conjuntiva**
  - $(\neg p \vee q) \wedge (q \rightarrow p)$ **no es forma normal conjuntiva**
- **Relación entre formas normales:**
  - F **es forma normal conjuntiva** $\Longrightarrow$
    $\Longrightarrow$ F **es forma normal negativa**
- **Transformación a forma normal conjuntiva:**
  - **Procedimiento** FNC(F)
    1. **Transformación a forma normal negativa**
    2. **Interiorización de las disyunciones**
       $$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$$
       $$(p \wedge q) \vee r \equiv (p \vee r) \wedge (q \vee r)$$
  - **Ejemplos:**
    $\mathrm{FNC}(p \wedge (q \rightarrow r)) = p \wedge (\neg q \vee r)$
    $FNC(\neg(p \wedge (q \rightarrow r))) = (\neg p \vee q) \wedge (\neg p \vee \neg r)$
- **Propiedades:**
  - FNC(F) **termina siempre**
  - FNC(F) **es una forma normal conjuntiva**
  - $\mathrm{FNC}(F) \equiv F$

Transformación a cláusulas

- **Cláusula de una fórmula clausular:**
  - **Def.: Sea F una fórmula clausular. Entonces**
    Cláusula(F) =
    = $\{F\}$, si F es un literal;
    = Cláusula(F_1) $\cup$ Cláusula(F_2), si $F = (F_1 \vee F_2)$
  - **Ejemplos:**
    Cláusula(p) $= \{p\}$
    Cláusula($\neg p$) $= \{\neg p\}$
    Cláusula($(\neg p \vee r) \vee (\neg p \vee q)$) $= \{\neg p, q, r\}$
- **Cláusulas de una fórmula en forma normal conjuntiva:**
  - **Sea F una fórmula en forma normal conjuntiva.**
    Cláusulas–FNC(F) =
    = Cláusulas–FNC(F_1) $\cup$ Cláusulas–FNC(F_2),
    si $F = (F_1 \wedge F_2)$
    = {Cláusula(F)}, en caso contrario
  - **Ejemplos:**
    Cláusulas–FNC($p \wedge (\neg q \vee r)$) =
    = $\{\{p\}, \{\neg q, r\}\}$
    Cláusulas–FNC($(\neg p \vee q) \wedge (\neg p \vee \neg r)$) =
    = $\{\{\neg p, q\}, \{\neg p, \neg r\}\}$

Transformación a cláusulas

- **Cláusulas de una fórmula:**
  - **Def.:** $\text{Cláusulas}(F) = \text{Cláusulas–FNC}(FNC(F))$
  - **Ejemplos:**
    $\text{Cláusulas}(p \wedge (q \rightarrow r)) = \{\{p\}, \{\neg q, r\}\}$
    $\text{Cláusulas}(\neg(p \wedge (q \rightarrow r))) = \{\{\neg p, q\}, \{\neg p, \neg r\}\}$
- **Cláusulas de un conjunto de fórmulas:**
  - **Def.:** $\text{Cláusulas–conjunto}(S) = \bigcup\{\text{Cláusulas}(F) : F \in S\}$
  - **Ejemplos:**
    $\text{Cláusulas–conjunto}(\{p \rightarrow q, q \rightarrow r\}) = \{\{\neg p, q\}, \{\neg q, r\}\}$
    $\text{Cláusulas–conjunto}(\{p \rightarrow q, q \leftrightarrow p\}) = \{\{\neg p, q\}, \{\neg q, p\}\}$

Transformación a cláusulas con OTTER

- **Cláusulas de una fórmula:**
  - **Entrada** `ej-fnc-fla.in`**:**

```
formula_list(sos).
p & (q -> r).
end_of_list.
```

  - **Salida:**

```
-------> sos clausifies to:
list(sos).
1 [] p.
2 [] -q|r.
end_of_list.
```

- **Cláusulas de un conjunto de fórmulas:**
  - **Entrada** `ej-fnc-conj.in`**:**

```
formula_list(sos).
p <-> q.
q -> r.
end_of_list.
```

  - **Salida:**

```
-------> sos clausifies to:
list(sos).
1 [] -p|q.
2 [] p| -q.
3 [] -q|r.
end_of_list.
```

Transformación a cláusulas con OTTER

- **Problema de los animales:**
  - **Entrada** (`ej-animales.in`)

```
formula_list(sos).
tiene_pelos | da_leche -> es_mamifero.
es_mamifero & (tiene_pezuñas | rumia) -> es_ungulado.
es_ungulado & tiene_cuello_largo -> es_jirafa.
es_ungulado & tiene_rayas_negras -> es_cebra.
tiene_pelos & tiene_pezuñas & tiene_rayas_negras.
-es_cebra.
end_of_list.
```

  - **Salida:**

```
-------> sos clausifies to:
list(sos).
1 [] -tiene_pelos|es_mamifero.
2 [] -da_leche|es_mamifero.
3 [] -es_mamifero| -tiene_pezugnas|es_ungulado.
4 [] -es_mamifero| -rumia|es_ungulado.
5 [] -es_ungulado| -tiene_cuello_largo|es_jirafa.
6 [] -es_ungulado| -tiene_rayas_negras|es_cebra.
7 [] tiene_pelos.
8 [] tiene_pezugnas.
9 [] tiene_rayas_negras.
10 [] -es_cebra.
end_of_list.
```

Símbolos proposicionales de cláusulas

- **Símbolos proposicionales de un literal:**
  - **Definición:**
    símbolos–proposicionales–literal(p) = $\{p\}$
    símbolos–proposicionales–literal($\neg p$) = $\{p\}$
- **Símbolos proposicionales de una cláusula:**
  - **Definición:**
    símbolos–proposicionales–cláusula(C) =
    = $\bigcup${símbolos–proposicionales–literal(L) : $L \in C$}
  - **Ejemplo:**
    símbolos–proposicionales–cláusula($\{p, q, \neg p\}$) = $\{p, q\}$
- **Símbolos proposicionales de un conjunto de cláusulas:**
  - **Definición:**
    símbolos–proposicionales–conjunto–cláusulas(S) =
    = $\bigcup${símbolos–proposicionales–cláusula(C) : $C \in S$}
  - **Ejemplo:**
    símbolos–proposicionales–conjunto–cláusulas
    ($\{\{p, q\}, \{\neg q, r\}\}$) = $\{p, q, r\}$

Interpretaciones de cláusulas

- **Interpretaciones de una cláusula:**
  - **Def.: I interpretación de C syss**
    I : símbolos–proposicionales–cláusula(C) $\rightarrow \{0,1\}$
  - **Def.: interpretaciones–cláusula(C) =**
    $= \{I : I$ interpretación de $C\}$
  - **Ej.: interpretaciones–cláusula($\{p, q, \neg p\}$) $= \{I_1, I_2, I_3, I_4\}$**

	p	q
I_1	0	0
I_2	0	1
I_3	1	0
I_4	1	1

- **Interpretaciones como conjuntos**
  - **Ej.: interpretaciones–cláusula($\{p, q, \neg p\}$) $= \{I_1, I_2, I_3, I_4\}$**

	p	q	
I_1	0	0	$\emptyset$
I_2	0	1	$\{q\}$
I_3	1	0	$\{p\}$
I_4	1	1	$\{p, q\}$

  - **Def.: interpretaciones–cláusula(C) =**
    $= \{I : I$ interpretación de $C\}$

Interpretaciones de cláusulas

- **Interpretaciones de conjuntos de cláusulas:**
  - **Definición:**
    I interpretación de S syss
    $I \subseteq$ símbolos–proposicionales–conjunto–cláusulas(S)
  - **Def.: interpretaciones–conjunto–cláusulas(C) =**
    $= \{I : I$ interpretación de $S\}$
  - **interpretaciones–conjunto–cláusulas($\{p, \neg q\}, \{\neg p, q\}\}$) =**
    $= \{\emptyset, \{p\}, \{q\}, \{p, q\}\}$

Modelos de cláusulas

- **Modelo de literal:**
  - **Definición:**
    $I \models p \iff p \in I$
    $I \models \neg p \iff p \notin I$
- **Modelo de una cláusula:**
  - **Def.:** $I \models C$ **syss** I **es modelo de algún literal de** C
  - **Ejemplos:**

$\{p, r\}$	es modelo de	$\{p, \neg q\}$
$\{r\}$	es modelo de	$\{p, \neg q\}$
$\{q, r\}$	no es modelo de	$\{p, \neg q\}$
$\{p, r\}$	no es modelo de	$\square$

- **Propiedad (Relación entre modelos de fórmulas clausulares y de cláusulas)**
  - **Sea** F **una fórmula clausular. Entonces**
    $I \models F \iff I \models$ **cláusula**(F)
- **Modelos de una cláusula:**
  - **Def.:** Modelos$(C) = \{I$ **interpretación de** $C : I \models C\}$
  - **Ejemplos:**
    Modelos$(\{\neg p, q\}) = \{\emptyset, \{q\}, \{p, q\}\}$
    Modelos$(\{\neg p, p\}) = \{\emptyset, \{p\}\}$
    Modelos$(\square) = \emptyset$

Modelos de cláusulas

- **Propiedad (Relación entre modelos de fórmulas clausulares y de cláusulas)**
  - Sea F una fórmula clausular. Entonces $\text{Modelos}(F) = \text{Modelos}(\text{cláusula}(F))$
- **Modelo de un conjunto de cláusulas:**
  - Def.: $I \models S$ syss I es modelo de todas las cláusulas de S
  - Ejemplos:

$\{p, r\}$	es modelo de	$\{\{p, \neg q\}, \{r\}\}$
$\{p\}$	no es modelo de	$\{\{p, \neg q\}, \{r\}\}$
$\{p, r\}$	es modelo de	$\square$

  - Relación entre modelos de fórmulas y de cláusulas: $I \models F \iff I \models \text{Cláusulas}(F)$
- **Modelos de un conjunto de cláusulas:**
  - Def.: $\text{Modelos}(S) = \{I \text{ interpretación de } S : I \models S\}$
  - Ejemplos:

    $\text{Modelos}(\{\{\neg p, q\}, \{\neg q, p\}\}) = \{\emptyset, \{p, q\}\}$
    $\text{Modelos}(\{\{\neg p, p\}, \{p\}, \{\neg q\}\}) = \emptyset$
    $\text{Modelos}(\{\{p, \neg p, q\}\}) = \{\emptyset, \{p\}, \{q\}, \{p, q\}\}$

  - Propiedad: $\text{Modelos}(F) = \text{Modelos}(\text{Cláusulas}(F))$

Cláusulas válidas y satisfacibles

- **Cláusulas válidas:**
  - **Def.: de cláusula válida:**
    $\models C$ syss toda interpretación de C es modelo de C
  - **Ejemplos:**
    $\{p, q, \neg p\}$ es válida.
    $\{p, q, \neg r\}$ no es válida.
  - **Propiedad: $\models C$ syss C contiene un literal y su complementario.**
  - **Propiedad: $\square$ no es válida**
  - **Propiedad: Sea F una fórmula clausular. Entonces**
    $\models F \iff \models$ Cláusulas(F)
- **Cláusula insatisfacible:**
  - **Def.: C es insatisfacible syss C no tiene modelo**
  - **Ejemplo: $\square$ es insatisfacible.**
  - **Propiedad: C es insatisfacible syss $C = \square$**
- **Cláusula satisfacible:**
  - **Def.: C es satisfacible syss C tiene modelo**
  - **Ejemplo: $\{p, \neg q\}$ es satisfacible**
  - **Propiedad: C es satisfacible syss $C \neq \square$**

Conjuntos inconsistentes de cláusulas

- **Conjunto válido de cláusulas:**
  - **Def.: S es válido ($\models S$) syss**
    todas las interpretaciones de S son modelos de S
  - **Ejemplos:**
    $\{\{\neg p, q\}, \{\neg q, p\}\}$ no es válido
    $\{\{\neg p, p\}, \{\neg q, q\}\}$ es válido
    $\emptyset$ es válido.
  - **Propiedad: $\models S$ syss todas las cláusulas de S son válidas**
- **Conjunto consistente de cláusulas:**
  - **Def.: S es consistente syss S tiene modelo**
  - **Def.: S es inconsistente syss S no tiene modelo**
  - **Ejemplo:**
    $\{\{\neg p, q\}, \{\neg q, p\}\}$ es consistente
    $\{\{\neg p, q\}, \{p\}, \{\neg q\}\}$ es inconsistente
- **Propiedad:**
  - **Sea S un conjunto de fórmulas. Entonces**
    S es consistente syss Cláusulas(S) es consistente

Consecuencia mediante cláusulas

- **Decisión de validez por cláusulas (sintáctico)**
  - **Propiedad: $\models F$ syss $\models$ Cláusulas(F)**
- **Consecuencia lógica entre cláusulas:**
  - **$S_1 \models S_2$ syss los modelos de S_1 son modelos de S_2**
  - **Ejemplos:**
    $\{\{\neg p, q\}, \{\neg q, r\}\} \models \{\{\neg p, r\}\}$
    $\{\{p\}\} \not\models \{\{p\}, \{r\}\}$
- **Propiedad: (Consecuencia entre fórmulas)**
  - **Sea S un conjunto de fórmulas. Entonces,**
    $S \models F \iff$ Cláusulas(S) $\models$ Cláusulas(F)
    $\iff$ Cláusulas($S \cup \{\neg F\}$) es inconsistente

Idea de inconsistencia por resolución

- Propiedades:
  - La cláusula vacía es insatisfacible
  - Un conjunto de cláusulas S es inconsistente syss la cláusula vacía es consecuencia de S
- Decisión de inconsistencia como búsqueda:
  - Para determinar si S es inconsistente, generar consecuencias de S hasta que se obtenga la cláusula vacía

Regla de resolución proposicional

- **Reglas de inferencia**
  - **Modus Ponens:**

```
p -> q          {-p,q}
p               {p}
------          ------
q               {q}
```

  - **Modus Tollens:**

```
p -> q          {-p,q}
-q              {-q}
------          ------
-p              {-p}
```

  - **Encadenamiento:**

```
p -> q          {-p,q}
q -> r          {-q,r}
------          ------
p -> r          {-p,r}
```

- **Regla de resolución proposicional:**

```
{p1,..., r,...,pm}
{q1,...,-r,...,qn}
--------------------
{p1,...,pm,q1,...,qn}
```

Regla de resolución proposicional

- **Resolvente proposicional:**
  - **Def.: Sean C_1 una cláusula, L un literal de C_1 y C_2 una cláusula que contiene el complementario de L. La resolvente de C_1 y C_2 respecto de L es**
    $$\text{resolvente}(C_1, C_2, L) = (C_1 - \{L\}) \cup (C_2 - \{\overline{L}\})$$
  - **Ejemplo:** $\text{resolvente}(\{p, q\}, \{\neg q, r\}, q) = \{p, r\}$
- **Resolventes de dos cláusulas:**
  - **Def.:** $\text{resolventes}(C_1, C_2)$ **es el conjunto de las resolventes entre C_1 y C_2**
  - **Ejemplos:**
    $$\begin{aligned} \text{resolventes}(\{\neg p, q\}, \{p, \neg q\}) &= \{\{p, \neg p\}, \{q, \neg q\}\} \\ \text{resolventes}(\{\neg p, q\}, \{p, q\}) &= \{\{q\}\} \\ \text{resolventes}(\{\neg p, q\}, \{q, r\}) &= \emptyset \end{aligned}$$
  - **Nota:** $\square \notin \text{resolventes}(\{p, q\}, \{\neg p, \neg q\})$
- **Adecuación de la regla de resolución:**
  - $C \in \text{resolventes}(C_1, C_2) \Longrightarrow \{C_1, C_2\} \models C$
- **CNS de inconsistencia**
  - **Sean S un conjunto de cláusulas, $C_1, C_2 \in S$ y $C \in \text{resolventes}(C_1, C_2)$. Entonces**
    S es inconsistente syss $S \cup \{C\}$ es inconsistente

Demostraciones por resolución

- **Ejemplo de refutación por resolución**
  - $\{\{p, q\}, \{\neg p, q\}, \{p, \neg q\}, \{\neg p, \neg q\}\} \vdash_{res} \square$

```
1 NIL {P,Q}
2 NIL {-P,Q}
3 NIL {P,-Q}
4 NIL {-P,-Q}
5 (2 1) {Q}
7 (4 3) {-Q}
8 (7 5) {}
```

- **Demostración por resolución:**
  - **Def.: La sucesión $(C_1, \ldots, C_n)$ es una demostración por resolución de C a partir de S si $C = C_n$ y para todo $i \in \{1, ..., n\}$ se verifica una de las siguientes condiciones:**
    *** $C_i \in S$;**
    *** existen $j, k < i$ tales que $C_i \in$ resolventes(C_j, C_k)**
  - **Def.: C es demostrable por resolución a partir de S si existe una demostración por resolución de C a partir de S**
  - **Representación: $S \vdash_{res} C$**

Demostraciones por resolución

- **Refutación por resolución:**
  - **Def.: La sucesión $(C_1, \ldots, C_n)$ es una refutación por resolución de S si es una demostración por resolución de la cláusula vacía a partir de S**
  - **Def.: S es refutable por resolución si existe una refutación por resolución a partir de S**
  - **Representación: $S \vdash_{res} \square$**
- **Propiedades del cálculo por resolución:**
  - **Adecuación: $S \vdash_{res} \square$ syss S es inconsistente**
  - **Completitud: S es inconsistente syss $S \vdash_{res} \square$**

Búsqueda de la prueba

- **Refutabilidad como búsqueda en espacios de estados**
- **Procedimiento de búsqueda de la prueba por saturación**
  - **Descripción**

```
* Entrada: S, un conjunto de cláusulas.
* Salida: Una refutación de S, si S es inconsistente;
          "S es consistente", en caso contrario.
* Procedimiento:
  1. Sea el SOPORTE el conjunto S
     y USABLES el conjunto vacío.
  2. Mientras que el SOPORTE sea no vacío,
     3. Sea ACTUAL la primera cláusula del SOPORTE,
     4. Añadir ACTUAL al principio de USABLES.
     5. Quitar ACTUAL del SOPORTE.
     6. Sea NUEVAS las resolventes de ACTUAL con
        las cláusulas de USABLES.
     7. Si una de las cláusulas de NUEVAS es la
        cláusula vacía, entonces devolver la
        prueba y terminar.
     8. Añadir las NUEVAS al final del SOPORTE.
  3. Si el SOPORTE es vacío, devolver que S es
     consistente.
```

Búsqueda de la prueba

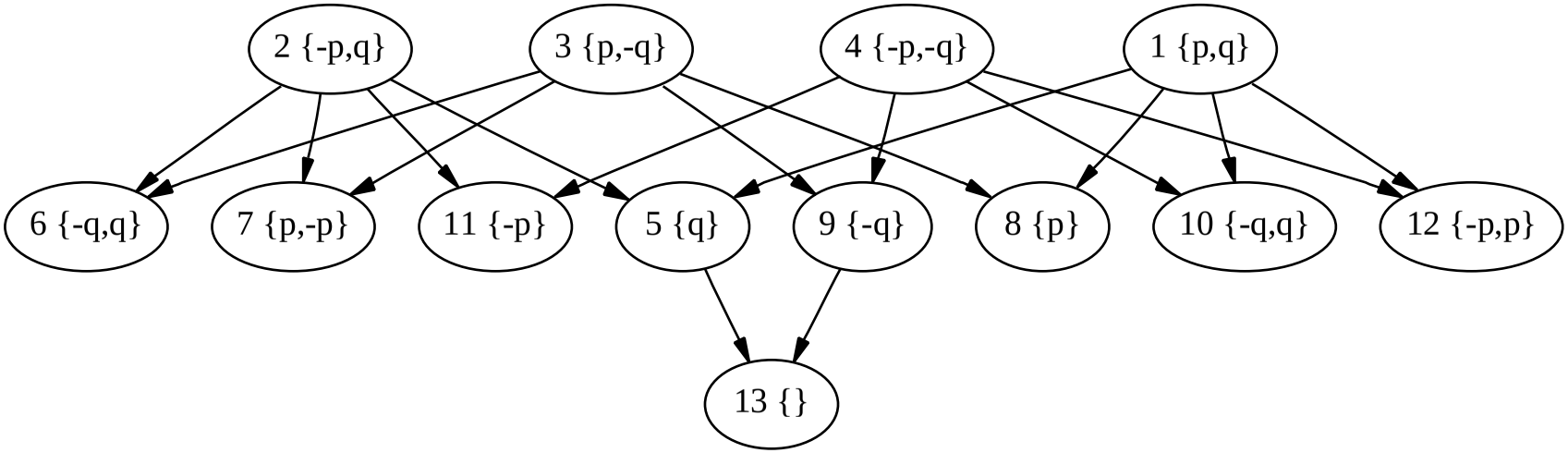
- **Ejemplo de búsqueda**
  - $S = \{\{p, q\}, \{\neg p, q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}$
  - **Por saturación**

```
========== Soporte ==========
1 Premisa {p,q}
2 Premisa {-p,q}
3 Premisa {p,-q}
4 Premisa {-p,-q}
====== Fin del soporte ======

1 Premisa {p,q}
2 Premisa {-p,q}
  ** 5 (2 1) {q}
3 Premisa {p,-q}
  ** 6 (3 2) {-q,q}
  ** 7 (3 2) {p,-p}
  ** 8 (3 1) {p}
4 Premisa {-p,-q}
  ** 9 (4 3) {-q}
  ** 10 (4 1) {-q,q}
  ** 11 (4 2) {-p}
  **    (4 1) {-p,p}
5 (2 1) {q}
  **    (5 4) {-p}
  **    (5 3) {p}
```

Búsqueda de prueba con eliminaciones

```
6 (3 2) {-q,q}
  **    (6 6) {q,-q}
  **    (6 5) {q}
  **    (6 2) {q,-p}
  **    (6 1) {q,p}
  **    (6 6) {-q,q}
  **    (6 4) {-q,-p}
  **    (6 3) {-q,p}
7 (3 2) {p,-p}
  **    (7 7) {-p,p}
  **    (7 4) {-p,-q}
  **    (7 2) {-p,q}
  **    (7 7) {p,-p}
  **    (7 3) {p,-q}
  **    (7 1) {p,q}
8 (3 1) {p}
  **    (8 7) {p}
  **    (8 4) {-q}
  **    (8 2) {q}
9 (4 3) {-q}
  **    (9 6) {-q}
  ** 12 (9 5) {}

========== Prueba ==========
 1 Premisa   {p,q}
 2 Premisa   {-p,q}
 3 Premisa   {p,-q}
 4 Premisa   {-p,-q}
 5 (2 1)     {q}
 9 (4 3)     {-q}
12 (9 5)     {}
===== Fin de la prueba =====
```

Grafo de la prueba



Búsqueda de prueba con eliminaciones

- **Cláusulas tautológicas**
  - **Def.: C es tautología syss C contiene un literal y su complementario**
  - **Ejemplos:**
    $\{p, q, \neg p\}$ es tautología
    $\{p, q, \neg r\}$ no es tautología
  - **C es tautología syss C es válida**
  - **Si $C \in S$ y C es tautología, entonces**
    S es inconsistente syss $S - \{C\}$ es inconsistente
- **Cláusulas unitarias:**
  - **Def.: C es unitaria syss C tiene sólo un literal.**

Búsqueda de prueba con eliminaciones

- **Procedimiento**

```
* Entrada: S, un conjunto de cláusulas.
* Salida: Una refutación de S, si S es inconsistente;
          NIL, en caso contrario.
* Procedimiento:
  1. Sea el SOPORTE el conjunto S y
     USABLES el conjunto vacío.
  2. Mientras que el SOPORTE sea no vacío y
     no se tenga una refutación,
     3. Sea ACTUAL la primera cláusula del SOPORTE,
     4. Añadir ACTUAL al principio de USABLES.
     5. Quitar ACTUAL del SOPORTE.
     6. Sea NUEVAS las resolventes de ACTUAL con
        las cláusulas de USABLES, que no sean
        tautologías ni estén en las USABLES o en
        el SOPORTE.
     7. Para cada cláusula C de NUEVAS,
        8.  Añadir C al final del SOPORTE.
        9.  Si C es la cláusula vacía,
            escribir la refutación y terminar.
        10. Si C es unitaria y su complementaria
            está en las USABLES o en el SOPORTE,
            escribir la refutación y terminar.
```

- **Propiedades de los procedimientos de prueba:**
  - **Terminación**
  - **Corrección**

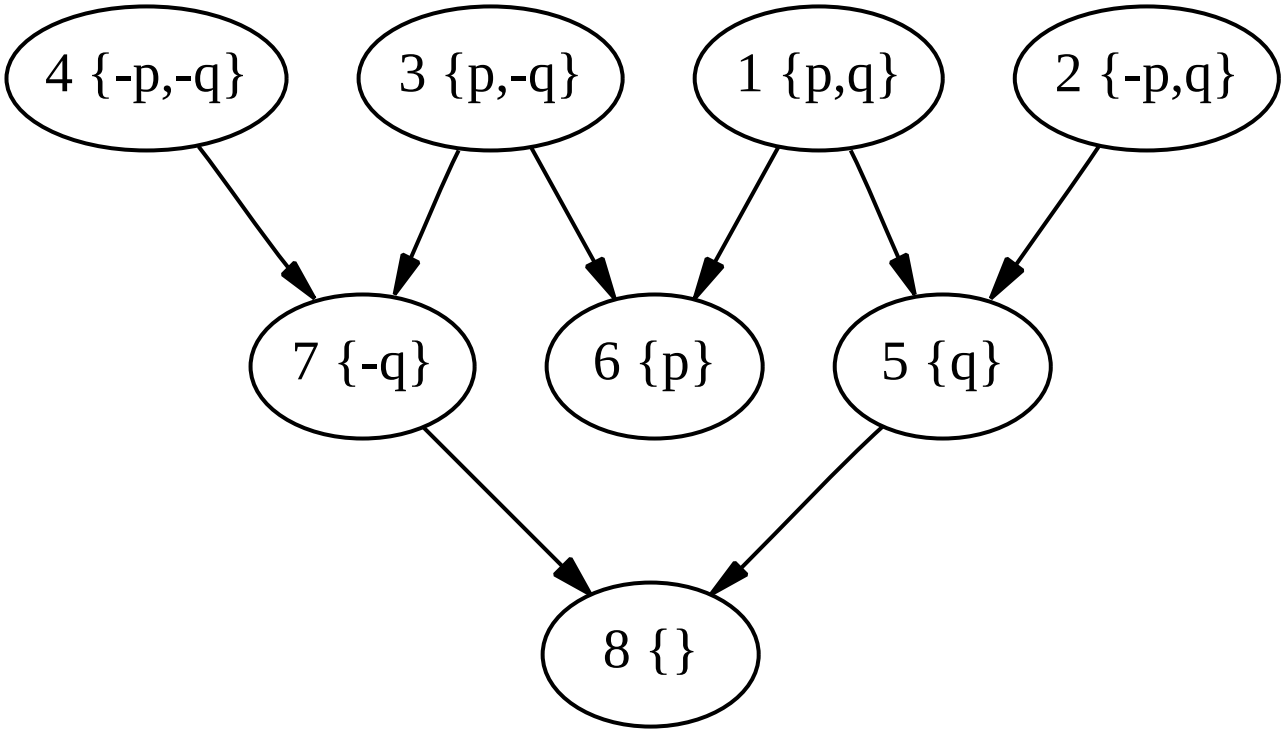
Búsqueda de prueba con eliminaciones

- **Ejemplo de búsqueda**
  - $S = \{\{p, q\}, \{\neg p, q\}, \{p, \neg q\}, \{\neg p, \neg q\}\}$
  - **Por saturación con eliminaciones**

```
========== Soporte ==========
1 NIL {P,Q}
2 NIL {-P,Q}
3 NIL {P,-Q}
4 NIL {-P,-Q}
====== Fin del soporte ======

1 NIL {P,Q}
2 NIL {-P,Q}
  ** 5 (2 1) {Q}
3 NIL {P,-Q}
  ** 6 (3 1) {P}
4 NIL {-P,-Q}
  ** 7 (4 3) {-Q}
8 (7 5) {}

========== Prueba ==========
1 NIL {P,Q}
2 NIL {-P,Q}
3 NIL {P,-Q}
4 NIL {-P,-Q}
5 (2 1) {Q}
7 (4 3) {-Q}
8 (7 5) {}
===== Fin de la prueba =====

T
```

Búsqueda de prueba con eliminaciones

- **Grafo de la prueba**



Búsqueda de prueba con OTTER

- **Entrada**

```
list(sos).
p | q.
-p | q.
p | -q.
-p | -q.
end_of_list.

set(binary_res).
set(sos_queue).
clear(back_sub).
clear(unit_deletion).
set(very_verbose).
```

Búsqueda de prueba con OTTER

- **Salida**

```
list(sos).
1 [] p|q.
2 [] -p|q.
3 [] p| -q.
4 [] -p| -q.
end_of_list.

========== start of search ==========
given clause #1: (wt=2) 1 [] p|q.

given clause #2: (wt=2) 2 [] -p|q.
  0 [binary,2.1,1.1] q|q.
** KEPT (pick-wt=1): 5 [binary,2.1,1.1,factor_simp] q.

given clause #3: (wt=2) 3 [] p| -q.
  0 [binary,3.1,2.1] -q|q.
  0 [binary,3.2,2.2] p| -p.
  0 [binary,3.2,1.2] p|p.
** KEPT (pick-wt=1): 6 [binary,3.2,1.2,factor_simp] p.

given clause #4: (wt=2) 4 [] -p| -q.
  0 [binary,4.1,3.1] -q| -q.
** KEPT (pick-wt=1): 7 [binary,4.1,3.1,factor_simp] -q.
--> UNIT CONFLICT at 0.01 sec -> 8 [binary,7.1,5.1] $F.
```

Búsqueda de prueba con OTTER

```
Length of proof is 2.  Level of proof is 1.

---------------- PROOF ----------------

1 [] p|q.
2 [] -p|q.
3 [] p| -q.
4 [] -p| -q.
5 [binary,2.1,1.1,factor_simp] q.
7 [binary,4.1,3.1,factor_simp] -q.
8 [binary,7.1,5.1] $F.

------------ end of proof -------------

============ end of search ============

-------------- statistics -------------
clauses given                  4
clauses generated              5
tautologies deleted            2
factor simplifications         3
clauses kept                   3
empty clauses                  1
usable size                    4
sos size                       3

----------- times (seconds) -----------
user CPU time          0.01
system CPU time        0.00
```

Búsqueda de prueba con OTTER

- **Opciones por defecto de OTTER**
  - **Peso**
  - **Subsunción hacia atrás**
- **Búsqueda con las opciones de OTTER**
  - **Entrada:**

```
list(sos).
p | q.
-p | q.
p | -q.
-p | -q.
end_of_list.

set(binary_res).
set(very_verbose).
```

Búsqueda de prueba con OTTER

- **Salida**

```
list(sos).
1 [] p|q.
2 [] -p|q.
3 [] p| -q.
4 [] -p| -q.
end_of_list.

========== start of search ==========
given clause #1: (wt=2) 1 [] p|q.

given clause #2: (wt=2) 2 [] -p|q.
  0 [binary,2.1,1.1] q|q.
** KEPT (pick-wt=1): 5 [binary,2.1,1.1,factor_simp] q.
5 back subsumes 2.
5 back subsumes 1.

given clause #3: (wt=1) 5 [binary,2.1,1.1,factor_simp] q.

given clause #4: (wt=2) 3 [] p| -q.
  0 [binary,3.2,5.1] p.
** KEPT (pick-wt=1): 6 [binary,3.2,5.1] p.
6 back subsumes 3.

given clause #5: (wt=1) 6 [binary,3.2,5.1] p.

given clause #6: (wt=2) 4 [] -p| -q.
  0 [binary,4.1,6.1] -q.
** KEPT (pick-wt=1): 7 [binary,4.1,6.1] -q.

--> UNIT CONFLICT at 0.01 sec --> 8 [binary,7.1,5.1] $F.
```

Búsqueda de prueba con OTTER

```
Length of proof is 2.  Level of proof is 1.

---------------- PROOF ----------------

1 [] p|q.
2 [] -p|q.
3 [] p| -q.
4 [] -p| -q.
5 [binary,2.1,1.1,factor_simp] q.
7 [binary,4.1,3.1,factor_simp] -q.
8 [binary,7.1,5.1] $F.

------------ end of proof -------------

============ end of search ============

-------------- statistics -------------
clauses given                  6
clauses generated              3
tautologies deleted            0
factor simplifications         1
clauses kept                   3
empty clauses                  1
clauses back subsumed          3
usable size                    3
sos size                       1

----------- times (seconds) -----------
user CPU time          0.01
system CPU time        0.00
```

Prueba de validez mediante resolución

- **Reducción de validez a refutación por resolución**
  - **Propiedad:** $\models F$ **syss Cláusulas**$(\neg F) \vdash_{res} \square$
- **Ejemplo de validez por resolución**
  - **Ejemplo:** $\models ((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$
  - **Entrada:**

```
formula_list(sos).
-(((p -> q) & (q -> r)) -> (p -> r)).
end_of_list.

set(binary_res).
```

  - **Salida**

```
-------> sos clausifies to:
list(sos).
1 [] -p|q.
2 [] -q|r.
3 [] p.
4 [] -r.
end_of_list.

---------------- PROOF ----------------
1 [] -p|q.
2 [] -q|r.
3 [] p.
4 [] -r.
5 [binary,1.1,3.1] q.
6 [binary,2.1,5.1] r.
7 [binary,6.1,4.1] $F.
------------ end of proof -------------
```

Consecuencia mediante resolución

- **Reducción de consecuencia a refutación por resolución**
  - **Sea S un conjunto de fórmulas y F una fórmula. Entonces $S \models F$ syss Cláusulas$(S \cup \{\neg F\}) \vdash_{res} \square$**
- **Ejemplo de consecuencia mediante resolución**
  - **$\{p \leftrightarrow q, r \leftrightarrow (p \wedge q)\} \models p \leftrightarrow r$**
  - **Entrada:**

```
formula_list(sos).
p <->q.
r <-> (p & q).
-(p <-> r).
end_of_list.
```

  - **Salida:**

```
-------> sos clausifies to:
list(sos).
1 [] -p|q.
2 [] p| -q.
3 [] -r|p.
4 [] -r|q.
5 [] r| -p| -q.
6 [] p|r.
7 [] -p| -r.
```

Consecuencia mediante resolución

```
========== start of search ==========
given clause #1: (wt=2) 1 [] -p|q.

given clause #2: (wt=2) 2 [] p| -q.
  0 [binary,2.1,1.1] -q|q.
  0 [binary,2.2,1.2] p| -p.

given clause #3: (wt=2) 3 [] -r|p.
  0 [binary,3.2,1.1] -r|q.
  Subsumed by 4.

given clause #4: (wt=2) 4 [] -r|q.
  0 [binary,4.2,2.2] -r|p.
  Subsumed by 3.

given clause #5: (wt=2) 6 [] p|r.
  0 [binary,6.1,1.1] r|q.
** KEPT (pick-wt=2): 8 [binary,6.1,1.1] r|q.
  0 [binary,6.2,4.1] p|q.
** KEPT (pick-wt=2): 9 [binary,6.2,4.1] p|q.
  0 [binary,6.2,3.1] p|p.
** KEPT (pick-wt=1): 10 [binary,6.2,3.1,factor_simp] p.
10 back subsumes 9.
10 back subsumes 6.
10 back subsumes 3.
10 back subsumes 2.

given clause #6: (wt=1) 10 [binary,6.2,3.1,factor_simp] p.
  0 [binary,10.1,1.1] q.
** KEPT (pick-wt=1): 11 [binary,10.1,1.1] q.
11 back subsumes 8.
11 back subsumes 4.
11 back subsumes 1.
```

Consecuencia mediante resolución

```
given clause #7: (wt=1) 11 [binary,10.1,1.1] q.

given clause #8: (wt=2) 7 [] -p| -r.
  0 [binary,7.1,10.1] -r.
** KEPT (pick-wt=1): 12 [binary,7.1,10.1] -r.
12 back subsumes 7.

given clause #9: (wt=1) 12 [binary,7.1,10.1] -r.

given clause #10: (wt=3) 5 [] r| -p| -q.
  0 [binary,5.1,12.1] -p| -q.
** KEPT (pick-wt=0):
        13 [binary,5.1,12.1,unit_del,10,11] $F.
--> EMPTY CLAUSE at 0.02 sec
    --> 13 [binary,5.1,12.1,unit_del,10,11] $F.

---------------- PROOF ----------------
1 [] -p|q.
3 [] -r|p.
5 [] r| -p| -q.
6 [] p|r.
7 [] -p| -r.
10 [binary,6.2,3.1,factor_simp] p.
11 [binary,10.1,1.1] q.
12 [binary,7.1,10.1] -r.
13 [binary,5.1,12.1,unit_del,10,11] $F.
------------ end of proof -------------
```

- **Comentarios**
  - **Subsunción hacia adelante**
  - **Eliminación unitaria**

Consecuencia mediante resolución

- **Ejemplo de no–consecuencia**
  - $\{p \leftrightarrow q, r \leftrightarrow (p \wedge q)\} \not\models p \wedge r$
  - **Entrada:**

```
formula_list(sos).
p <->q.
r <-> (p & q).
-(p & r).
end_of_list.

set(binary_res).
set(very_verbose).
```

Consecuencia mediante resolución

```
-------> sos clausifies to:

list(sos).
1 [] -p|q.
2 [] p| -q.
3 [] -r|p.
4 [] -r|q.
5 [] r| -p| -q.
6 [] -p| -r.
end_of_list.
======= end of input processing =======

=========== start of search ===========
given clause #1: (wt=2) 1 [] -p|q.

given clause #2: (wt=2) 2 [] p| -q.

given clause #3: (wt=2) 3 [] -r|p.

given clause #4: (wt=2) 4 [] -r|q.

given clause #5: (wt=2) 6 [] -p| -r.
** KEPT (pick-wt=1):
        7 [binary,6.1,3.2,factor_simp] -r.
7 back subsumes 6.
7 back subsumes 4.
7 back subsumes 3.

given clause #6: (wt=1)
                 7 [binary,6.1,3.2,factor_simp] -r.
```

Consecuencia mediante resolución

```
given clause #7: (wt=3) 5 [] r| -p| -q.
** KEPT 8 [binary,5.1,7.1] -p| -q.
** KEPT 9 [binary,5.2,2.1,unit_del,7,factor_simp] -q.
** KEPT 10 [binary,5.3,1.2,unit_del,7,factor_simp] -p.
8 back subsumes 5.
9 back subsumes 8.
9 back subsumes 2.
10 back subsumes 1.

given clause #8: (wt=1) 9  -q.

given clause #9: (wt=1) 10  -p.

Search stopped because sos empty.
```

- **Problema de los animales con OTTER**
  - **Tema 1 pp. 8–12**

Bibliografía

- Burris, S.N. *Logic for Mathematics and Computer Science* (Prentice–Hall, 1998)
  - Cap. 2 "Propositional logic"
- Chang, C.L. y Lee, R.C.T *Symbolic Logic and Mechanical Theorem Proving* (Academic Press, 1973)
  - Cap. 2 "The propositional logic"
  - Cap. 5 "The resolution principle"
- Genesereth, M.R. *Computational Logic* (27 March 2000)
  - Cap. 5 "Propositional resolution"
- Nilsson, N.J. *Inteligencia artificial (Una nueva síntesis)* (McGraw–Hill, 2000)
  - Cap. 14 "La resolución en el cálculo proposicional"